# WEEK - 13

**Aim:** to analyze and understand the architecture and working principle of a pre-trained deep learning model (such as VGG16, ResNet, inception) used for image classification and feature extraction.

**Algorithm:**

1) import a pre-trained model from keras applications.
2) load model with pretrained weights
3) display model summary (layers, parameters)
4) Analyze layer types (conv, pool, Dense)
5) Identify trainable vs non-trainable parameters
6) optionally, visualize feature map.

**pseudo code:**

Begin

Import deep learning library (eg:- Tensorflow, Pytorch)

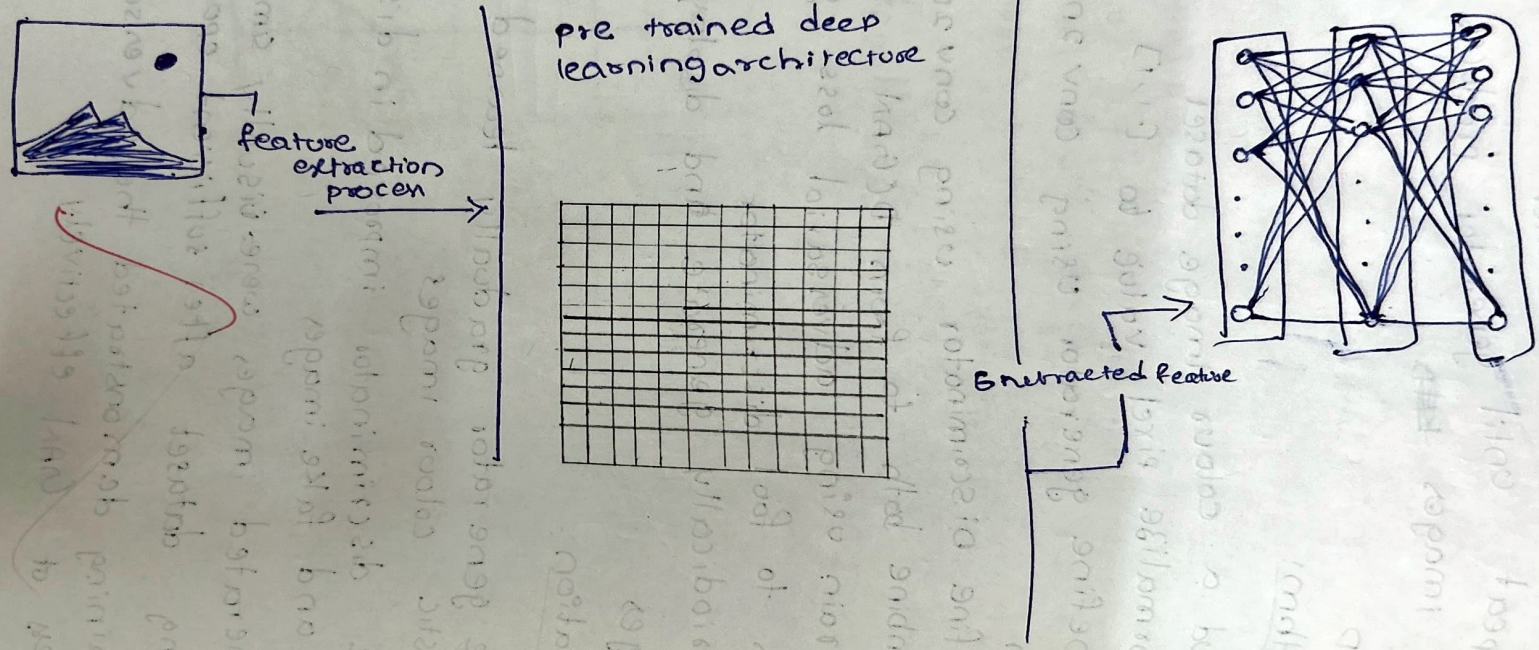
load a pre trained model.
- → include weights = 'imagenet'
- → exclude top layer if using for feature extraction.

Display model summary
- → print layer names, types, output shapes & parameter counts

for each layer in model:
- → Identify type (conv, pool, Dense, etc---)
- → Note activation function and No. of. filters visualize architecture diagram
- → show flow from input image to output class train the model

Freeze lower layer if needed.

# Usage of PRE-trained Architecture.



Observation:

→ the pre trained model has convolutional, pooling and fully connected layers

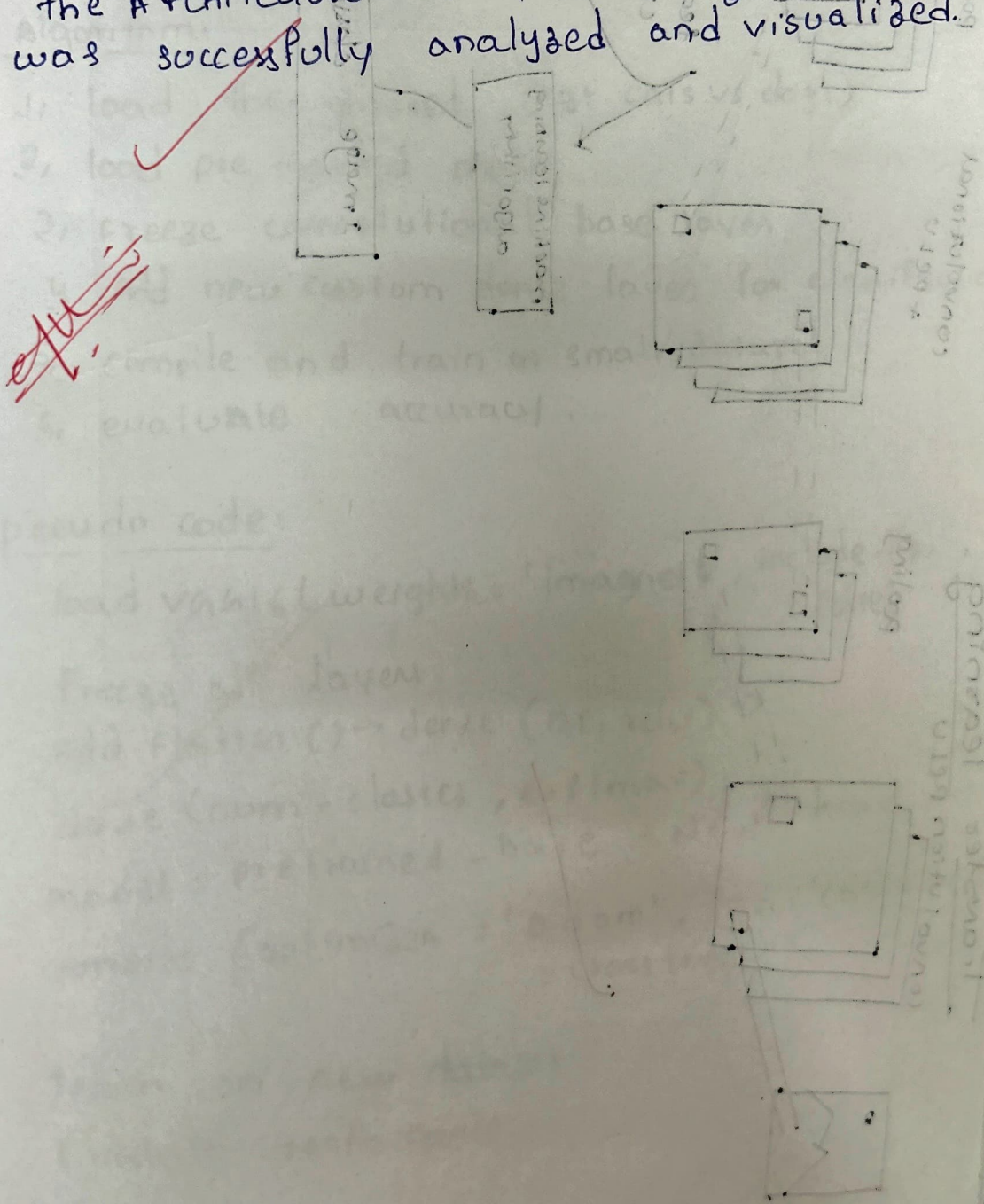
→ convolutional layer extract features, pooling layers reduce size.

→ pre trained weights allow faster training and better accuracy

→ sample images are correctly classified (or) have meaningful feature extracted

Result:

the Architecture of pre-trained CNN's was successfully analyzed and visualized.



Output

Total parameters : 138357544

Trainable parameters: 138357544

Non Trainable parameters : 0

layers Names:

convol : convo2D

bn1 : Batch Normail 2D

maxpool : Relu

layer1 : sequantial

layer2 : sequential

layer3 : sequential

layer4: sequantial

avgpool : adaptive avg pool2D

fc : linear.

```python
# Lab 13: Understanding the architecture of a pre-trained model (VGG16)

import tensorflow as tf
from tensorflow.keras.applications import VGG16

# Load pre-trained VGG16 model with ImageNet weights
model = VGG16(weights="imagenet", include_top=True)

# Display model summary
model.summary()

# Plot model architecture
tf.keras.utils.plot_model(model, to_file="vgg16_architecture.png", show_shapes=True, show_layer_names=True)

print("\n✅ Model architecture plot saved as 'vgg16_architecture.png'")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 ━━━━━━━━━━━━━━━━━━━━ 3s 0us/step
Model: "vgg16"
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 ━━━━━━━━━━━━━━━━━━━━ 3s 0us/step
Model: "vgg16"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer_13 (InputLayer) | (None, 224, 224, 3) | 0 |
| block1_conv1 (Conv2D) | (None, 224, 224, 64) | 1,792 |
| block1_conv2 (Conv2D) | (None, 224, 224, 64) | 36,928 |
| block1_pool (MaxPooling2D) | (None, 112, 112, 64) | 0 |
| block2_conv1 (Conv2D) | (None, 112, 112, 128) | 73,856 |
| block2_conv2 (Conv2D) | (None, 112, 112, 128) | 147,584 |
| block2_pool (MaxPooling2D) | (None, 56, 56, 128) | 0 |
| block3_conv1 (Conv2D) | (None, 56, 56, 256) | 295,168 |
| block3_conv2 (Conv2D) | (None, 56, 56, 256) | 590,080 |
| block3_conv3 (Conv2D) | (None, 56, 56, 256) | 590,080 |
| block3_pool (MaxPooling2D) | (None, 28, 28, 256) | 0 |
| block4_conv1 (Conv2D) | (None, 28, 28, 512) | 1,180,160 |
| block4_conv2 (Conv2D) | (None, 28, 28, 512) | 2,359,808 |
| block4_conv3 (Conv2D) | (None, 28, 28, 512) | 2,359,808 |
| block4_pool (MaxPooling2D) | (None, 14, 14, 512) | 0 |
| block5_conv1 (Conv2D) | (None, 14, 14, 512) | 2,359,808 |
| block5_conv2 (Conv2D) | (None, 14, 14, 512) | 2,359,808 |
| block5_conv3 (Conv2D) | (None, 14, 14, 512) | 2,359,808 |
| block5_pool (MaxPooling2D) | (None, 7, 7, 512) | 0 |
| flatten (Flatten) | (None, 25088) | 0 |
| fc1 (Dense) | (None, 4096) | 102,764,544 |
| fc2 (Dense) | (None, 4096) | 16,781,312 |
| predictions (Dense) | (None, 1000) | 4,097,000 |

Total params: 138,357,544 (527.79 MB)
Trainable params: 138,357,544 (527.79 MB)
Non-trainable params: 0 (0.00 B)

✅ Model architecture plot saved as 'vgg16_architecture.png'